

VizMark: Benchmarking Visibility Preprocessing

Piotr Dubla, Tai Lucas, Sebastian Murray-Roberts
{pdubla, tlucas, smurray-}@cs.uct.ac.za

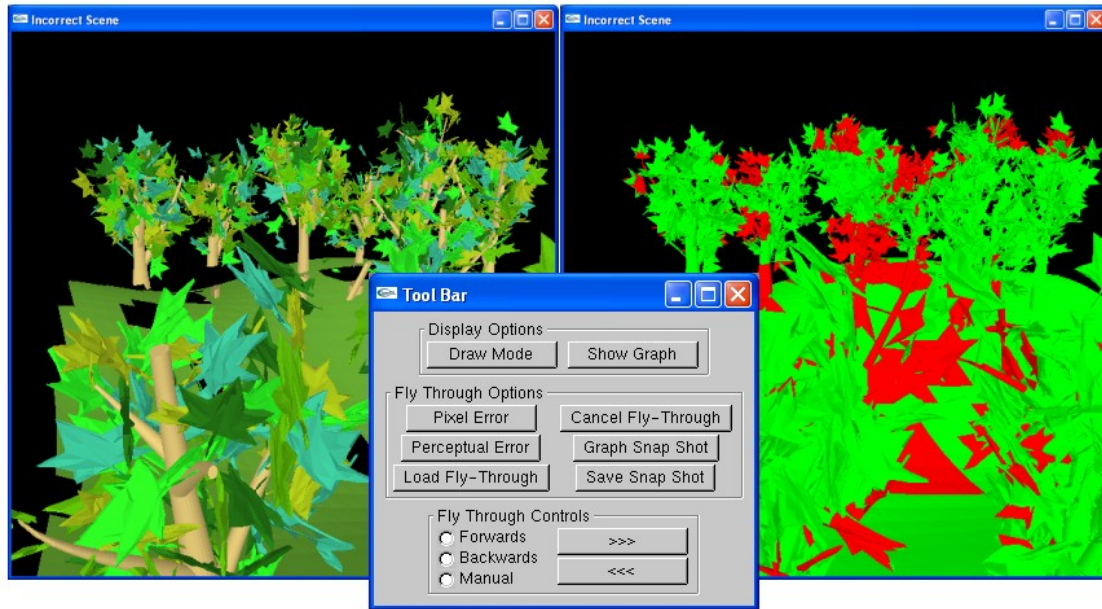


Figure 1: Benchmarker, toolbar (centre), perceptual error (left) and pixel error (right)

ABSTRACT

We present a new means of comparing visibility algorithms by means of the implementation of a standard reference solution against which visibility algorithms may be tested. This will allow new and existing visibility algorithms to be objectively tested.

An accurate reference solution was developed that employs an optimised ray casting algorithm to calculate visibility. Due to the excessive computational overhead in this calculations, a parallel implementation reduces the amount of time needed to produce the reference solution.

The benchmarker component determines the accuracy of the algorithm undergoing testing based on a number of image error metrics which take into account the quality of the final, rendered image.

This paper discusses the components that make up the VizMark system and how each is tested.

Keywords: visibility, ray casting, parallel computing, error metrics, rendered image quality

1. INTRODUCTION

The VizMark system aims to provide a platform upon which authors of visibility algorithms can test ideas in a standardised environment. It is hoped that this will provide authors of such algorithms with a method for discarding unpromising ideas early, or refining promising ones. This should streamline the development of algorithms leading to faster and more accurate algorithms.

The VizMark system is unique in that it not only measures the accuracy of the algorithm being tested, in terms of the number of correctly or incorrectly classified polygons, but also on the basis of the quality of the final rendered image. The rationale behind this is that an algorithm may classify some visible polygons in a scene as being invisible. However, the overall image quality may still remain high. This means that an inaccurate solution may still be written so as to produce a high quality rendered result.

The aim of the VizMark project is to develop a highly accurate visibility solution for a 3D scene

which is then compared against a test algorithm. The reference solution will be time consuming to run for large scenes and will need to be parallelised to reduce this time. VizMark also aims to provide a standard set of tests that will provide the author of a test implementation with an array of diverse and detailed information regarding how inaccurate their algorithm is, and where it is inaccurate.

2. IMPLEMENTATION

• Test Data

The test data for the following sections will be four scenes, referred to as Rocket, Forest, Hill Top and Large Forest. Table 1 shows a rendered preview of each of the scenes along with the corresponding polygon count.


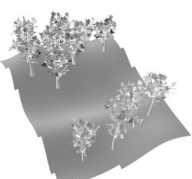

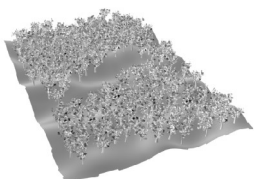
Rocket	Forest
	
11,396	243,660
Hill Top	Large Forest
	
1,145,662	4,003,262

Table 1: Test data

The Rocket and Hill Top scenes have high levels of occlusion in the scene because many of the constituent polygons are contained within some structure. For instance, the hull of the rocket model occludes any internal features, and the buildings in the Hill Top scene contain many unseen, smaller features. The Forest scenes, however, have a lower level of occlusion as there are more possible paths through the small polygons that make up the leaves. Much of the scene is composed of densely packed polygons, visible from most points of view.

2.1. REFERENCE SOLUTION

The final solution consists of a complete program that takes parameters such as subdivision granularity, grid size and a model file then generates a potential

visibility set (PVS) that contains visibility information for each region in the scene.

The reference solution is broken down into three core components, region setup, polygon classification and traversal.

• Scene Partitioning

This partitions the scene along each axis, according to three parameters. The subsequent traversal stage then computes visibility from each of these regions.

• Polygon Classification to Regions

The naïve approach for classifying the polygons in a scene would be for every region to intersect every polygon in the scene, with the bounding box of the region. This technique slows down substantially as the number of regions and/or the number of polygons in the scene increases.

For the partitioning algorithm, the structure in scene files is exploited to gain a substantial performance increase without the need for developing any extra techniques. This is done by comparing the object axis-aligned bounding boxes (AABB) in the scene against the AABB of each region. These AABB-overlap tests reduce the search space and increase the performance of the partitioning process.

Partitioning of forest scene		
Subdivisions	Without AABB tests	With AABB tests
10x10x10	55.33 sec	1.63 sec
9x9x9	44.42 sec	0.97 sec
8x8x8	24.77 sec	0.57 sec
7x7x7	16.85 sec	0.51 sec

Table 2: Polygon classification speed-up

• Region Traversal

The traversal is the core component of the solution and makes use of data generated by region partitioning and calculates PVS data for the scene.

The core of the algorithm is based around ray casting. Visibility is determined by casting a number of rays between two regions that are being tested. Visibility is determined between a pair of faces on two regions, a vector is then calculated between points on each respective region face, using a *jittered grid* approach, illustrated below. A jittered approach was necessary to provide better coverage of the space between the faces. In the figures below, jittering is explained the

difference between jittered and non-jittered ray firing is demonstrated.

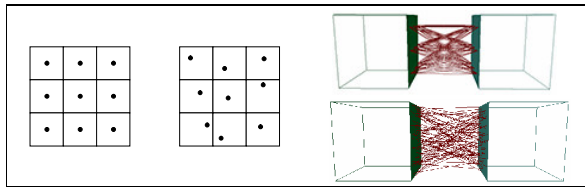


Figure 2: A region face. A non-jittered (left) and jittered (right) grid comparison.

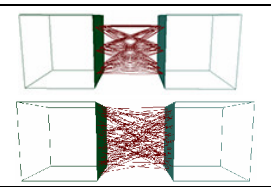


Figure 3: Rays being cast between a non-jittered (above) and jittered (below) grid.

2.2. BENCHMARKER

The benchmarker makes use of a number of perceptual quality metrics that measure impact that the culled polygons will have on the final, rendered image.

Perceptual quality metrics are used to determine the quality of an image, when compared to a reference image. In general, there are three types of metrics which can be used to determine image quality:

1. *Spatial domain metrics* compare images at the pixel level, giving information about the structural similarity between images. These metrics generally focus on the difference between pixel colour values in each image.
2. *Spatial-Frequency domain metrics* give information about the changes of image intensity and contrast by examining the frequency of colour values across the image.
3. *Human Vision System (HVS) metrics* can be a combination of the above two and are designed to determine the quality of an image based on how it would be perceived by a human observer. These techniques rely on models of the human visual system and are used to determine the amount of realism an image portrays.

- **Choice of Error Metrics**

It is important to design a metric, or set of metrics that provides a rich and diverse range of information regarding the errors contained in an image. In general, there is no, single metric that can accurately measure all errors in a satisfactory manner. It is therefore necessary to develop a set of metrics that together, comprise a more complete metric.

The simplest metric is the pixel-level error metric, which measures the differences between two images, with correct polygons drawn in green and incorrect polygons drawn in red. The principle behind this metric is that, while polygons in incorrectly classified regions are drawn in red, these polygons may not all be visible from a given point of view. Conversely, even if a polygon is visible, it may not be completely visible from a given point of view. The viewer may, for instance, be facing away from the region. While the pixel-level error merely counts the number of red pixels on the screen, it provides information regarding the extent to which incorrectly classified polygons are actually visible.

The second metric computed by the bench marker is the perceptual error metric. This metric is computed with the model rendered in its original colours or with any textures that have been associated with it. The perceptual metric is designed to measure the amount of error that can be seen in the image, having taken into account the colouring of objects in the scene, as well as the effects of lighting on those objects. The perceptual metric is composed of two metrics. The L^2 mean squared error [1,3,4,5,6], a spatial domain metric, computes structural differences between the two images by calculating the sum of the squared differences between the pixel colour values in each image. This reports on any, differences between the images, on a global scale. The second metric, differences near areas of high transition (DNAHT) [1], is based on the fact that the human visual system is sensitive to sharp changes in contrast or colour. This metric determines which areas in the reference image have high transitions between pixel values, masking off these areas (see Figure 4 below). Based on the masked areas, the metric finds discontinuities in the test image.

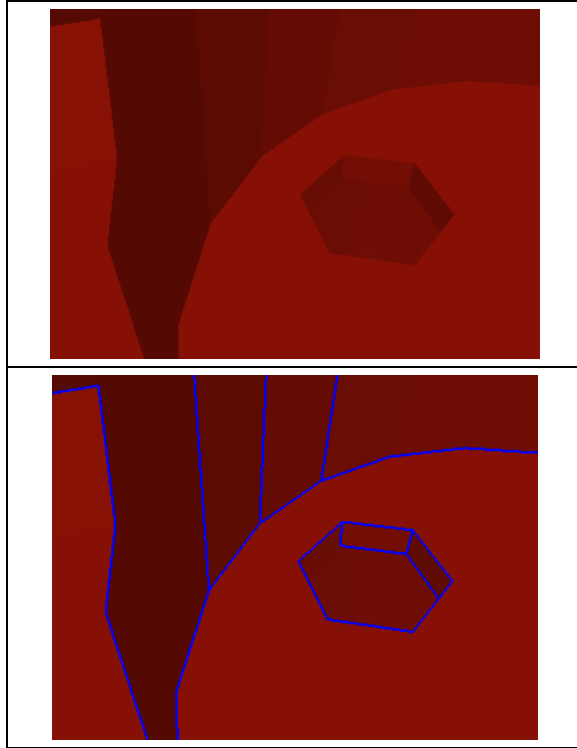


Figure 4: Original image (above) showing transition lines. Blue, lines (below) mask off areas of high transition.

These metrics, when combined, provide a wide spectrum of information regarding the difference between the two images.

The benchmarker accepts as input two files containing the PVS data of the standard solution and the test algorithm. The benchmarker is comprised of three main components:

1. The raw pixel-level error metrics, which compare the two images and provide information about the number of incorrect pixels and whether the pixels are clustered together in groups called connected regions [2]. The metrics for this component are calculated based on the fact that all correctly classified polygons are coloured in green, and all incorrectly classified polygons are coloured in red. This allows the benchmarker to determine to what extent incorrect polygons are indeed visible from each point of view.
2. The perceptual error metrics compare the images produced by the two algorithms when rendered in their original colours. The perceptual error metrics used in the benchmarker determine both the structural error in the final images as well as

a measure of how the human visual system would interpret the error in the image.

3. The final output, a fly-through of the scene, is used to generate a report of the error data. This takes the form of a graph representing the error at each frame of the fly-through and a summary of the error statistics recorded over the entire fly-through.

The figure below illustrates typical output from the benchmarker.

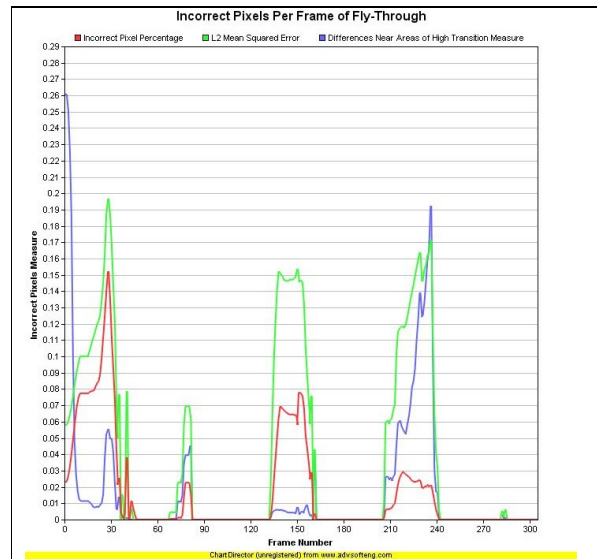


Figure 5: All three metrics plotted on same set of axes

3. RESULTS

3.1. REFERENCE SOLUTION

After theoretical analysis of the algorithm, the complexity was calculated to be $O(N^2G^4HP + N^3)$ where N is the number of regions, G is the grid size, H is the average number of regions that a ray passes through and P the average amount of polygons per region.

The table below illustrates how the number of regions into which the scene is divided affects the number of regions removed from the visibility calculations. For each region that is considered in the traversal algorithm, the total cost of the computation is dependent on the number of other regions that need to be considered. The higher the subdivision, the more costly the overall computation however, unnecessary calculations are avoided by not considering invisible regions.

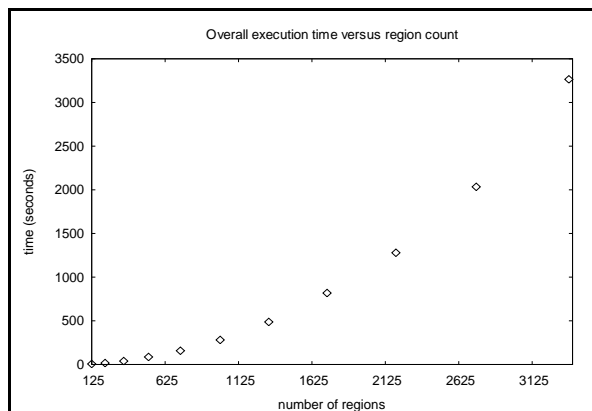
Subdivision		Rocket	Forest	Hilltop
	2x2x2	0.00%	0.00%	0.00%
	3x3x3	14.81%	7.41%	7.41%
	4x4x4	25.00%	10.94%	20.31%
	5x5x5	32.80%	17.60%	32.00%
	6x6x6	39.81%	23.15%	39.35%
	10x10x10	84.90%	60.20%	75.20%

Table 3: Correlation between subdivision size and the mean percentage of regions culled.

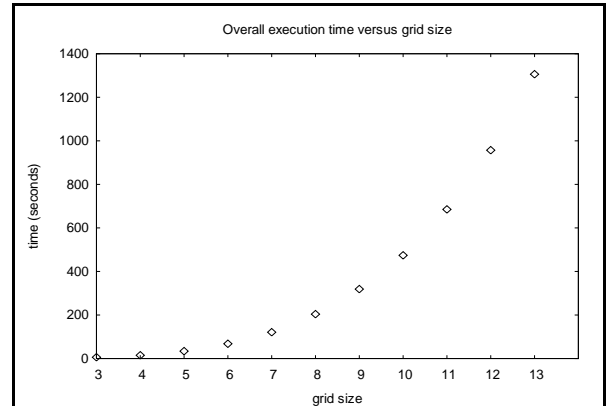
The two parameters that affect the performance of the traversal algorithm are *region count* and *grid size*. The traversal stage of the program occupies a large proportion of the total execution time of the program and is most affected by the changes in these variables.

The combination of the number of regions and the grid size affect the overall accuracy of the implementation or *sampling density*. The sampling density is related to the distance between the grid points on the faces of the regions. Even if the grid size is small, the grid points may be closely spaced because the region sizes are small. As the number of regions in the scene increases, the area of each of the faces decreases and so the grid points on the face are placed closer together.

The timing data for a fixed grid size (as with the previous tests) and variable region partitions, ranging from 125 regions to 3375 regions is plotted in Graph 1. The second graph shows timing data for a variable grid sizes, ranging from 3 to 13. The timing values are taken for the fully parallel implementation, running on all nodes.



Graph 1: Execution time with varying region counts



Graph 2: Execution time with varying grid sizes

The execution time increases in a non-linear fashion, as either of these parameters is increased. Due to the nature of the traversal algorithm, the amount of computation rises rapidly, as there is an exponential relation between computation cost and the two parameters.

Although it may appear as if these two parameters cause a compound loss in performance, it is important to note that they work together toward the accuracy of the solution. Once these parameters have been set though, the system will scale in a linear fashion as the number of nodes in the cluster is increased.

The tables below compare the overall execution time of the visibility preprocessor for both the serial and parallel implementations.

The number of polygons in each test scene has been chosen to properly represent an increasing amount of data for the preprocessor to compute.

Scene	Execution time (seconds)
Rocket	22.0
Forest	264.00
Hill Top	2917.00
Large Forest	11460.00

Table 4: Total execution time for serial program

Scene	Parallel Performance	
	time (seconds)	speedup
Rocket	5.29	4.16
Forest	63.48	4.16
Hill Top	755.73	3.86
Large Forest	2923.16	3.92

Table 5: Total execution time and speedup for parallel implementation

- **Overall Speedup**

Given linear scalability, the speedup in each test scene for an ideal parallel implementation would be five times that of the serial version (for five nodes running). In this implementation, the speedup ranges from 4.16 for the smallest scene, to 3.92 for the largest scene. The speedup is therefore not proportional to the number of nodes running on the cluster and indicates a loss of efficiency of the program due to parallelisation.

- **Test Scenes**

In the ideal case, while considering efficiency loss, the speedup should be the same for all scenes, which would indicate that the system is unaffected by the size of the input data. There is only a small effect on the overall speedup as a result of the scene size.

While the efficiency of the system is affected by the size of the input data set, this effect is only marginal when considering how the input size increases for each test scene and would suggest that overall scalability of the system is good for scenes of appreciable size.

3.2. BENCHMARKER

Benchmarker results were generated using an approximation of an aggressive visibility algorithm which removes more regions than an exact solution. This test PVS consisted of a PVS generated by the reference solution, with a pre-defined percentage of regions removed.

The results showed that the metrics were strongly correlated where large numbers of regions were incorrectly classified as invisible from certain point of view. This is due to the fact that each region that is not drawn results in a potentially significant amount of polygons missing from the image. Scenes in which the background colour showed through, such as the forest scene yielded higher values for the perceptual metrics than for the pixel metrics, as the sharp changes between colouring of the scene and the background accentuated the differences in the image. Scenes with enclosed areas such as the rocket yielded higher pixel error values, as the perceptual metrics are not sensitive to small changes in colour. In cases where large numbers of polygons were missing, the perceptual metrics did return higher values.

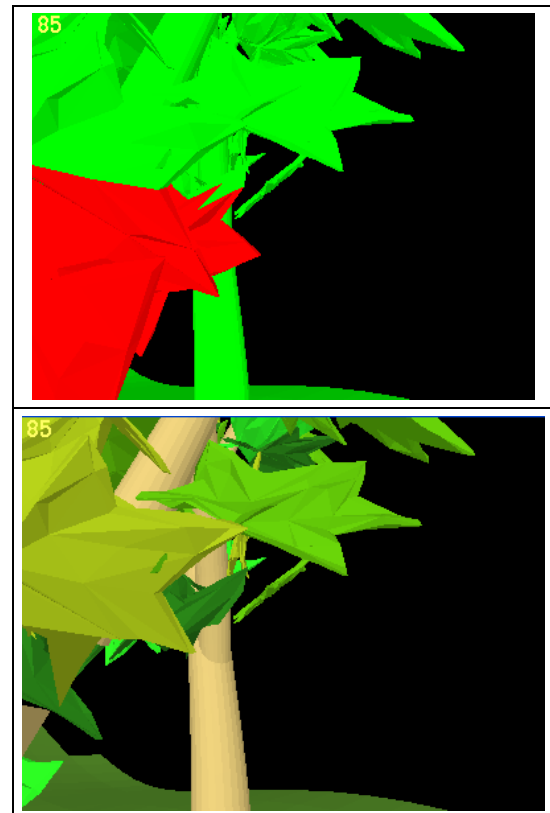


Figure 6: Similarities between metrics.

The images above show the error values for the forest scene. Due to the fact that one of the leaves, which is drawn in red, is missing, both the incorrect pixel count and the perceptual metrics found large errors for this frame.

The images below show the maximum error for the metrics using a different fly-through and highlight the differences between the metrics. The pixel error metric is sensitive to errors close to the viewpoint, due to the fact that it counts the number of red pixels. However, the perceptual error is sensitive to errors near areas of high transition or large structural differences. In reality the leaf in the pixel error image would not be seen, and this would not have a large impact on visual quality. However, the missing polygons of the tree shown highlighted by the perceptual metric would have a large impact on visual quality.

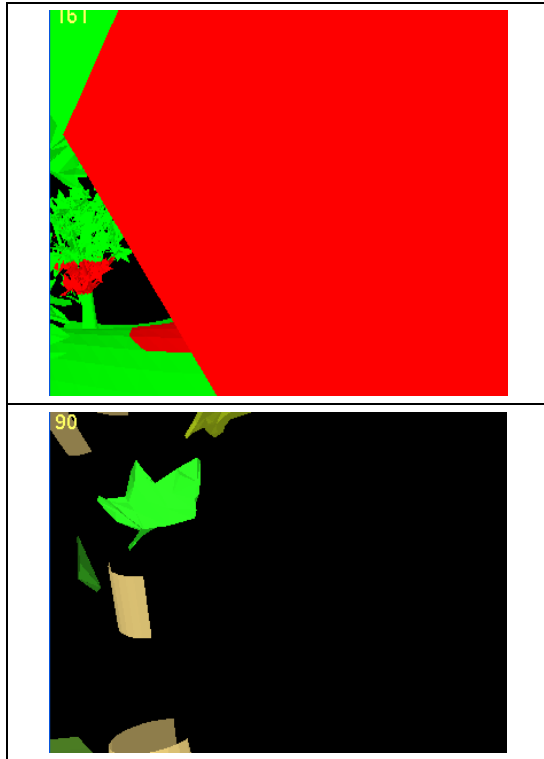


Table 6: Differences between metrics

4. CONCLUSIONS

The reference solution produces accurate results, even with modest values of the grid size and region count parameters. With high enough values of these parameters, this implementation produces near-exact visibility.

Visibility calculations have been shown to be computationally intensive for scenes of appreciable size and VizMark demonstrates that running such calculations in a distributed environment can significantly reduce the amount of time required. The system design shows good scalability as the problem size and number of nodes increases. Consistency checks of the system determined that the parallel program behaved in a deterministic way, thereby producing the same output for constant parameter settings yet varying node counts.

5. FUTURE WORK

- Implementation of a region to polygon visibility algorithm

The solution would greatly benefit from a change of the algorithm from a region to region to a region to

polygon visibility solution, where only the relevant polygons that are visible in each region are recorded.

- Ensure that the algorithm is exact, and that this can be verified.

The reference solution provides a means against which other algorithms are checked. It would greatly benefit from a traversal algorithm that could guarantee exact visibility for a given scene.

- Use of rational arithmetic

The accuracy of the solution would greatly benefit if a rational arithmetic is used. Numerical inaccuracies would be avoided entirely and this would be of most benefit for the many intersection tests. This would however have a drastic effect on the speed of the program and so the traversal algorithm would have to undergo significant optimisation before this approach is used.

- Parallel Implementation

Scalability of the parallel preprocessor has been tested here with only a small cluster size. The effectiveness of the strategies devised here ought to undergo testing with larger node counts so as to properly test this and to identify the susceptibility to existing bottlenecks.

- Benchmarker

Polygon cutting refers to the technique of dividing a single polygon into two or more polygons which comprise the original. Polygon cutting is particularly useful in from-region visibility as it prevents polygons that belong to more than one region from being coloured incorrectly. At present, the benchmarker draws incorrect regions after drawing the polygons in each region's visibility list. This is done to ensure that under-classified regions (conservative algorithms) are drawn in red. The idea behind polygon cutting in the benchmarker case is that polygons which belong to more than one region are divided along the boundary of the two regions. This ensures that only polygons which belong to a certain region are drawn if that region is classified as visible.

REFERENCES

- [1] Sahasrabudhe, N. "Structured Spatial Domain Image and Data Comparison Metrics". Proceedings of the conference on Visualization '99, pages 97-104, 1999

- [2] Nirenstien, S. "Fast and Accurate Visibility Preprocessing". PhD thesis, U. Cape Town, 2003.
- [3] Tian, Q., Xue, Q., Yu, L., Sebel, N., Huang, T.S. "Toward an improved error metric". IEEE International Conference on Image Processing, October 2004
- [4] Fienup, J.R.. "Invariant error metrics for image reconstruction". Applied Optics Vol. 36, No. 32, November 1997
- [5] Wang, Z., Simoncelli, E.P. "Stimulus Synthesis for Efficient Evaluation and Refinement of Perceptual Image Quality Metrics". Human Vision and Electronic Imaging IX. In *Proceedings of SPIE* volume 5292, January 2004
- [6] Zhou, H., Chen, M., Webster, M.F. "Comparative Evaluation of Visualization and Experimental Results Using Image Comparison Metrics". In *IEEE Visualisation* 2002, October – November 2002